**Development process**

|  |  |
| --- | --- |
| **Date** | **20-06-2025** |
| **Project Name** | **Empowering Career Progression with Tailored Cover Letters** |
| **Team ID** | **SWTID1749741458** |
| **Maximum Marks** | **5M** |

Project Title:

Smart AI Cover Letter Generator using Gemini API & Streamlit

Objective:

To develop an intelligent and interactive web application that generates personalized, professional cover letters based on either form input or uploaded resumes, using Google's Gemini large language model (LLM).

Tech Stack:

- Frontend & Web App Framework: Streamlit  
- AI Model: Google Generative AI (Gemini 1.5 Flash)  
- PDF Handling: PyPDF2  
- Environment Variable Management: python-dotenv  
- Programming Language: Python 3.8+

Directory Structure:

project\_root/  
├── app.py # Main Streamlit app with navigation  
├── homepage.py # UI for homepage  
├── form\_based.py # Form-based cover letter generator  
├── resume\_based.py # Resume-based cover letter generator  
├── .env # Environment file storing Gemini API key  
├── requirements.txt # Required Python libraries

How to Run the Project:

1. Clone or Download the Project:  
 git clone <repo\_url>  
 cd <project\_directory>  
  
2. Create a Virtual Environment:  
 python -m venv venv  
 source venv/bin/activate (or venv\Scripts\activate on Windows)  
  
3. Install Dependencies:  
 pip install -r requirements.txt  
  
4. Create the .env File:  
 GEMINI\_KEY="your\_gemini\_api\_key\_here"  
  
5. Run the App:  
 streamlit run app.py

Included Code Files:

app.py

# app.py  
import streamlit as st  
from homepage import show\_homepage  
from form\_based import form\_based\_cover\_letter  
from resume\_based import resume\_based\_cover\_letter  
  
st.set\_page\_config(page\_title="Smart Cover Letter Generator", page\_icon="💼")  
  
# Custom CSS for a modern, clean look  
st.markdown(  
 """  
 <style>  
 .main {background-color: #f8f9fa;}  
 .stButton>button {background-color: #4CAF50; color: white; width: 100%; margin-bottom: 10px;}  
 .stDownloadButton>button {background-color: #1976D2; color: white;}  
 .sidebar-title {font-size: 1.3rem; font-weight: 600; margin-bottom: 1.5rem;}  
 </style>  
 """,  
 unsafe\_allow\_html=True  
)  
  
# Initialize session state for navigation  
if 'page' not in st.session\_state:  
 st.session\_state['page'] = "Home"  
  
st.sidebar.markdown('<div class="sidebar-title">🧭 Navigation</div>', unsafe\_allow\_html=True)  
  
# Sidebar navigation with buttons (no radio or dropdown)  
if st.sidebar.button("🏠 Home", use\_container\_width=True):  
 st.session\_state['page'] = "Home"  
if st.sidebar.button("📝 Form-Based", use\_container\_width=True):  
 st.session\_state['page'] = "Form-Based"  
if st.sidebar.button("📄 Resume-Based", use\_container\_width=True):  
 st.session\_state['page'] = "Resume-Based"  
  
# Main area  
if st.session\_state['page'] == "Home":  
 show\_homepage()  
elif st.session\_state['page'] == "Form-Based":  
 form\_based\_cover\_letter()  
else:  
 resume\_based\_cover\_letter()

homepage.py

import streamlit as st  
   
def show\_homepage():  
 st.title("💼 Welcome to Smart AI Cover Letter Generator")  
  
 # Custom CSS for fun, modern cards and sidebar  
 st.markdown(  
 """  
 <style>  
 .option-card {  
 padding: 1.5rem;  
 margin-bottom: 1.5rem;  
 border-radius: 0.7rem;  
 background: linear-gradient(135deg, #e3f2fd 60%, #fffde7 100%);  
 box-shadow: 0 4px 16px rgba(0,0,0,0.07);  
 transition: transform 0.1s;  
 }  
 .option-card:hover {  
 transform: scale(1.03);  
 box-shadow: 0 8px 32px rgba(25, 118, 210, 0.12);  
 }  
 .option-title {  
 font-size: 1.3rem;  
 font-weight: bold;  
 color: #1976D2;  
 }  
 .option-desc {  
 font-size: 1.05rem;  
 color: #333;  
 margin-bottom: 1rem;  
 }  
 </style>  
 """, unsafe\_allow\_html=True  
 )  
  
 st.markdown("### 🚀 What would you like to do today?")  
  
 col1, col2 = st.columns(2)  
  
 with col1:  
 st.markdown('<div class="option-card">', unsafe\_allow\_html=True)  
 st.markdown('<span class="option-title">📝 Form-Based Cover Letter</span>', unsafe\_allow\_html=True)  
 st.markdown(  
 '<div class="option-desc">Manually enter your details<br>'  
 'Fine-tune every field<br>'  
 'Best for custom, detailed applications</div>',  
 unsafe\_allow\_html=True  
 )  
 if st.button("✨ Go to Form-Based"):  
 st.session\_state['page'] = "Form-Based"  
 st.markdown('</div>', unsafe\_allow\_html=True)  
  
 with col2:  
 st.markdown('<div class="option-card">', unsafe\_allow\_html=True)  
 st.markdown('<span class="option-title">📄 Resume-Based Cover Letter</span>', unsafe\_allow\_html=True)  
 st.markdown(  
 '<div class="option-desc">Upload your resume (PDF)<br>'  
 'Let AI extract your info<br>'  
 'Fastest way to get started</div>',  
 unsafe\_allow\_html=True  
 )  
 if st.button("🚀 Go to Resume-Based"):  
 st.session\_state['page'] = "Resume-Based"  
 st.markdown('</div>', unsafe\_allow\_html=True)  
  
 st.markdown("---")  
 st.markdown(  
 "<span style='font-size:1.1rem;'>Use the sidebar or the cards above to select your preferred method for generating a tailored cover letter.</span>",  
 unsafe\_allow\_html=True  
 )

form\_based.py

import streamlit as st  
import google.generativeai as genai  
import os  
from dotenv import load\_dotenv  
  
def form\_based\_cover\_letter():  
 load\_dotenv()  
 api\_key = os.getenv("GEMINI\_KEY")  
 if not api\_key:  
 st.error("❌ GEMINI\_KEY not found in .env file. Please set it.")  
 st.stop()  
  
 genai.configure(api\_key=api\_key)  
 model = genai.GenerativeModel('gemini-1.5-flash')  
  
 with st.form("cover\_letter\_form"):  
 st.subheader("Form-Based Cover Letter")  
 name = st.text\_input("Your Name")  
 job\_title = st.text\_input("Job Title")  
 primary\_skills = st.text\_area("Primary Skills (comma-separated)", placeholder="e.g. Python, Django, REST APIs")  
 experience = st.slider("Years of Experience", 0, 30, 3)  
 client\_job\_post = st.text\_area("Client's Job Post Description")  
 relevant\_projects = st.text\_area("Relevant Projects (comma-separated)")  
 client\_name = st.text\_input("Client Name")  
 start\_date = st.date\_input("Start Date")  
 deadline = st.date\_input("Deadline")  
 tone = st.selectbox("Preferred Tone", ["Formal", "Friendly", "Persuasive", "Confident"])  
 submit = st.form\_submit\_button("Generate Cover Letter")  
  
 if submit:  
 data = {  
 "name": name,  
 "jobTitle": job\_title,  
 "primarySkills": [s.strip() for s in primary\_skills.split(',')],  
 "experience": experience,  
 "clientJobPost": client\_job\_post,  
 "relevantProjects": [p.strip() for p in relevant\_projects.split(',')],  
 "clientName": client\_name,  
 "startDate": str(start\_date),  
 "deadline": str(deadline),  
 "tone": tone  
 }  
 clean\_job\_post = data['clientJobPost'].replace('\n', ' ')  
 prompt = f"""  
You are a skilled freelancer with expertise in writing cover letters that have helped secure jobs. Your task is to craft a compelling cover letter that persuades the client to contact the freelancer.  
  
The letter should focus on results and solving the client’s goals, not just earning money.  
  
Here is the candidate's information in JSON format:  
  
{{  
 "Name": "{data['name']}",  
 "JobTitle": "{data['jobTitle']}",  
 "PrimarySkills": {data['primarySkills']},  
 "Experience": {data['experience']},  
 "ClientJobPost": "{clean\_job\_post}",  
 "RelevantProjects": {data['relevantProjects']},  
 "ClientName": "{data['clientName']}",  
 "StartDate": "{data['startDate']}",  
 "Deadline": "{data['deadline']}",  
 "Tone": "{data['tone']}"  
}}  
  
Write a personalized, professional cover letter based on this.  
"""  
 try:  
 response = model.generate\_content(prompt)  
 st.success("Cover letter generated!")  
 st.text\_area("Your AI-Generated Cover Letter", response.text, height=350)  
 st.download\_button("Download Cover Letter", response.text, file\_name="cover\_letter.txt")  
 except Exception as e:  
 st.error(f"❌ Error: {e}")

resume\_based.py

import streamlit as st  
import google.generativeai as genai  
import os  
from dotenv import load\_dotenv  
from PyPDF2 import PdfReader  
  
def resume\_based\_cover\_letter():  
 load\_dotenv()  
 api\_key = os.getenv("GEMINI\_KEY")  
 if not api\_key:  
 st.error("❌ GEMINI\_KEY not found in .env file. Please set it.")  
 st.stop()  
  
 genai.configure(api\_key=api\_key)  
 model = genai.GenerativeModel('gemini-1.5-flash')  
  
 with st.form("resume\_cover\_letter\_form"):  
 st.subheader("Resume-Based Cover Letter")  
 resume\_file = st.file\_uploader("Upload your Resume (PDF)", type="pdf")  
 job\_description = st.text\_area("Client's Job Post Description")  
 tone = st.selectbox("Preferred Tone", ["Formal", "Friendly", "Persuasive", "Confident"])  
 submit\_resume = st.form\_submit\_button("Generate Cover Letter")  
  
 if submit\_resume:  
 if resume\_file is None or not job\_description.strip():  
 st.warning("Please upload your resume and fill in the job description.")  
 else:  
 try:  
 pdf\_reader = PdfReader(resume\_file)  
 resume\_text = "\n".join([page.extract\_text() for page in pdf\_reader.pages if page.extract\_text()])  
 resume\_text = resume\_text[:3000] # Limit for prompt size  
 clean\_job\_post = job\_description.replace('\n', ' ')  
 prompt = f"""  
You are a skilled freelancer with expertise in writing cover letters that have helped secure jobs. Your task is to craft a compelling cover letter that persuades the client to contact the freelancer.  
  
The letter should focus on results and solving the client’s goals, not just earning money.  
  
Here is the candidate's resume content (truncated if too long):  
  
---RESUME---  
{resume\_text}  
---END RESUME---  
  
Here is the job description:  
{clean\_job\_post}  
  
Preferred Tone: {tone}  
  
Write a personalized, professional cover letter based on this.  
"""  
 response = model.generate\_content(prompt)  
 st.success("Cover letter generated!")  
 st.text\_area("Your AI-Generated Cover Letter", response.text, height=350)  
 st.download\_button("Download Cover Letter", response.text, file\_name="cover\_letter.txt")  
 except Exception as e:  
 st.error(f"❌ Error: {e}")

requirements.txt

streamlit>=1.20.0  
google-generativeai>=0.3.0  
python-dotenv>=1.0.0  
PyPDF2>=3.0.0

.env

GEMINI\_KEY="YOUR\_GEMINI\_KEY"